Exhibit B

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP. INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 5,892,900

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
155.	Products infringing: Any product using Microsoft Product Activation or Reader Activation feature.
A virtual distribution environment comprising	
(a) a first host processing environment comprising	computer running a Microsoft product containing the Product Activation feature, including Windows XP, Office XP, Visio 2002. Reader using its activation feature.
(1) a central processing unit;	CPU of computer
(2) main memory operatively connected to said central processing unit;	main memory of computer
(3) mass storage operatively connected to said central processing unit and said main memory;	hard disk or other mass storage contained in computer
(b) said mass storage storing tamper resistant software designed to be loaded into said main memory and executed by said central processing unit, said tamper resistant software comprising:	Microsoft Product Activation software
(1) machine check programming which derives information from one or more aspects of said host processing environment,	Product Activation software generates hardware information relating to the host processing environment as part of the activation process
(2) one or more storage locations storing said information;	hardware information is stored in the computer's storage
(3) integrity programming which	
(i) causes said machine check programming to derive said information,	each time the Microsoft program starts up after initial activation, Product Activation checks the originally derived hardware information against current hardware
(ii) compares said information to information previously stored in said one or more storage locations, and	each time the Microsoft program starts up after initial activation, Product Activation checks the originally derived hardware information against current hardware
(iii) generates an indication based on the result of said comparison; and	Product Activation software indicates whether the test has passed or failed
(4) programming which takes one or more actions based on the state of said indication;	
(i) said one or more actions including at least temporarily halting further processing.	Product Activation software will allow system startup procedures to continue, if test succeeds, or discontinue startup and offer user opportunity to reactivate if the test fails

Exhibit B

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,892,900

156.	Product Infringing: Any product using Microsoft Product Activation or Reader Activation feature.
A virtual distribution environment comprising	
(a) a first host processing environment comprising	computer running a Microsoft product containing the Product Activation feature, including Windows XP, Office XP, Visio 2002 and Reader
(1) a central processing unit;	CPU of computer
(2) main memory operatively connected to said central processing unit;	main memory of computer
(3) mass storage operatively connected to said central processing unit and said main memory;	hard disk or other mass storage contained in computer
(b) said mass storage storing tamper resistant software designed to be loaded into said main memory and executed by said central processing unit, said tamper resistant software comprising:	Microsoft Product Activation software
(1) machine check programming which derives information from one or more aspects of said host processing environment,	Product Activation software generates hardware information relating to the host processing environment as part of the activation process
(2) one or more storage locations storing said information;	hardware information is stored in the computer's storage
(3) integrity programming which	
(i) causes said machine check programming to derive said information,	each time the Microsoft program starts up after initial activation, Product Activation checks the originally derived hardware information against current hardware
(ii) compares said information to information previously stored in said one or more storage locations, and	each time the Microsoft program starts up after initial activation, Product Activation checks the originally derived hardware information against current hardware
(iii) generates an indication based on the result of said comparison; and	Product Activation software indicates whether the test has passed or failed
(4) programming which takes one or more actions based on the state of said indication;	
(i) said one or more actions including at least temporarily disabling certain functions.	Product Activation may disable the underlying software from generating new files or running user applications if the test fails

Exhibit B

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,892,900

157.	Product Infringing: Any product using Microsoft Product Activation or Reader Activation feature.
A virtual distribution environment comprising	
(a) a first host processing environment comprising	computer running a Microsoft product containing the Product Activation feature, including Windows XP, Office XP, Visio 2002 and Reader
(1) a central processing unit;	CPU of computer
(2) main memory operatively connected to said central processing unit;	main memory of computer
(3) mass storage operatively connected to said central processing unit and said main memory;	hard disk or other mass storage contained in computer
(b) said mass storage storing tamper resistant software designed to be loaded into said main memory and executed by said central processing unit, said tamper resistant software comprising:	Microsoft Product Activation software
(1) machine check programming which derives information from one or more aspects of said host processing environment,	Product Activation software generates hash information relating to the host processing environment as part of the activation process
(2) one or more storage locations storing said information;	hardware information is stored in the computer's storage
(3) integrity programming which	
(i) causes said machine check programming to derive said information,	each time the Microsoft program starts up after initial activation, Product Activation checks the originally derived hardware information against current hardware
(ii) compares said information to information previously stored in said one or more storage locations, and	each time the Microsoft program starts up after initial activation, Product Activation checks the originally derived hardware information against current hardware
(iii) generates an indication based on the result of said comparison; and	Product Activation software indicates whether the test has passed or failed
(4) programming which takes one or more actions based on the state of said indication;	
(i) said one or more actions including displaying a message to the user.	Product Activation software displays a message to the user if the test fails

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.

INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,892,900

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
156.	Products infringing: Windows Media Player
A virtual distribution environment comprising	
a first host processing environment comprising	WMP with Individualized DRM client (referred to hereafter as the Individualized WMP) running on a client computer
a central processing unit	Client CPU
main memory operatively connected to said central processing unit	Client memory
mass storage operatively connected to said central processing unit and said main memory	Local disk drive
said mass storage storing tamper resistant software designed to be loaded into said main memory and executed by said central processing unit, said tamper resistant software comprising:	Individualized WMP (I-WMP) stored on disk and loaded into main memory upon execution. I-WMP is tamper resistant.
machine check programming which derives information from one or more aspects of said host processing environment,	Individualization module is generated by the MS individualization service either when the un-individualized WMP tries to open licensed content that requires a security upgrade (aka, Individualization) or when the user requests an upgrade un-provoked. The individualization module is unique and signed and is bound to a unique hardware ID using the MS machine activation process.
one or more storage locations storing said information	The aforementioned unique feature are located in multiple places or storage locations
integrity programming which	
causes said machine check programming to derive said information,	The ID is regenerated by WMP/DRM client when first loading the Individualized DRM Client to access a piece of content requiring the security upgrade.
compares said information to information previously stored in said one or more storage locations, and	The program checks the new copy against the one to which the Individualized DRM client is bound.
generates an indication based on the result of said comparison; and	Program stores the result of this check.
programming which takes one or more actions based on the state of said indication	If these are not equal, the user is notified via a message stating that he/she must acquire a security upgrade (that is, the current security upgrade is invalid). If they are equal then processing of songs requiring Individualization continues.
said one or more actions including at least temporarily disabling certain functions.	Songs targeted to this Individualization module cannot be accessed until the upgrade is correct.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,892,900

157. A virtual distribution environment comprising	Infringing products include: Windows Media Player
a first host processing environment comprising	See 156
a central processing unit	See 156
main memory operatively connected to said central processing unit	See 156
mass storage operatively connected to said central processing unit and said main memory	See 156
said mass storage storing tamper resistant software designed to be loaded into said main memory and executed by said central processing unit, said tamper resistant software comprising:	See 156
machine check programming which derives information from one or more aspects of said host processing environment,	See 156
one or more storage locations storing said information	See 156
integrity programming which causes said machine check programming to derive said information compares said information to information previously stored in said one or more storage locations, and	See 156
generates an indication based on the result of said comparison; and	See 156
programming which takes one or more actions based on the state of said indication	See 156
said one or more actions including displaying a message to the user.	If these are not equal, the user is notified via a message stating that he/she must acquire a security upgrade (that is, the current security upgrade is invalid).

Exhibit B

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 5,892,900

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
157.	Infringing Product: Microsoft's Windows File Protection and System File Checker features, embodied in Microsoft's Windows 2000, Windows XP products, and Server 2003
A virtual distribution environment comprising	
(a) a first host processing environment comprising	computer running Microsoft Windows 2000 or Windows XP.
(1) a central processing unit;	CPU of computer
(2) main memory operatively connected to said central processing unit;	main memory of computer
(3) mass storage operatively connected to said central processing unit and said main memory;	hard disk or other mass storage contained in computer
(b) said mass storage storing tamper resistant software designed to be loaded into said main memory and executed by said central processing unit, said tamper resistant software comprising:	Windows File Protection process/service ("WFP") and System File Checker (SFC.exe) features of winlogon.exe. Winlogon.exe is treated as a "critical" service by the Windows operating system. Files supporting WFP (including winlogon.exe, sfc.exe, sfc.dll (2000 only), sfcfiles.dll (2000 only) and sfc_os.dll (XP only)) are "protected" files and are signed using a signature verified by a hidden key. In Windows 2000, WFP uses hidden functions within the sfc.dll library. Functions are imported by "ordinal" instead of "name."
(1) machine check programming which derives information from one or more aspects of said host processing environment,	Winlogon either directly or using another dll (XP) or using SFC.dll (2000) determines if changed file was protected, computes the hash of protected files and, if necessary, computes the hash of the file in the dll cache before using it to replace a file overwritten by an incorrect version of the file.
(2) one or more storage locations storing said information;	hardware information is stored in the computer's memory
(3) integrity programming which	
(i) causes said machine check programming to derive said information,	Windows notifies Winlogon when there has been a system directory change or a change in the dll cache.
(ii) compares said information to information previously stored in said one or more storage locations, and	Winlogon either directly or using another dll (XP) or using SFC.dll (2000) compares computed hash with hash in the hash database created from the Catalog file(s), and, if there is a difference, compares the hash of the file in the dll cache to the hash database created from

Exhibit B
6

	the Catalog file(s) before using it to replace an overwritten file.
(iii) generates an indication based on the result of said comparison; and	An event is written to the Event Viewer if hashes do not agree.
(4) programming which takes one or more actions based on the state of said indication;	Depending on the circumstances, WFP displays several messages to the user, including prompting the user to contact the system administrator, and to insert a CD-ROM.
(i) said one or more actions including displaying a message to the user.	See above. Messages also constitute viewable Event Property pop-ups.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.

INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,917,912

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
6.	Product Infringing: XBox
A process comprising the following steps:	The process constitutes assembly and use of components making up an XBox game.
accessing a first record containing information directly or indirectly identifying one or more elements of a first component assembly,	The first record consists of the second file table on an XBox DVD. This table identifies the .xbe file which includes the game information.
at least one of said elements including at least some executable programming,	The xbe file includes executable programming.
at least one of said elements constituting a load module,	The xbe file is a load module.
said load module including executable programming and a header;	The xbe file includes a header.
at least a portion of said header is a public portion which is characterized by a relatively lower level of security protection; and	Most information the xbe header is not obfuscated.
at least a portion of said header is a private portion which is characterized, at least some of the time, by a level of security protection which is relatively higher than said relatively lower level of security protection,	The entry point address and the kernel image thunk address listed in the xbe header are obfuscated and therefore at a higher level of security protection.
using said information to identify and locate said one or more elements;	The second file table identifies the .xbe file, including where that file is located.
accessing said located one or more elements;	The .xbe file is accessed by the XBox.
securely assembling said one or more elements to form at least a portion of said first component assembly;	At runtime, the .xbe file is assembled with certain services of the operating system to form a component assembly. Security associated with this assembling process includes verifying signatures associated with portions of the .xbe file, and replacing obfuscated calls to operating system services with actual addresses. The assembly may also include patch files downloaded from a remote server.
executing at least some of said executable	Game play requires execution of the

Exhibit B

programming; and	assembled programming.
checking said record for validity prior to performing said executing step.	The second file table is protected by a digital signature, and is not loaded/used unless the digital signature is verified against the file.
7. A process as in claim 6 in which:	
said relatively lower level of security protection comprises storing said public header portion in an unencrypted state; and	The header is protected by the techniques protecting the xbe such as signing and security descriptors, but it is not encrypted except as noted below.
said relatively higher level of security protection comprises storing said private header portion in an encrypted state.	The entry point address and the kernel image thunk address listed in the xbe header are obfuscated. The Xbox SDK's (XDK) image build uses a key value shared with the retail XBox to perform two XOR operations against the addresses

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,917,912

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
8.	Infringing products: Microsoft CLR or CCLR and .NET Framework SDK and products that include one or both of these.
A process comprising the following steps:	
(a) accessing a first record containing information directly or indirectly identifying one or more elements of a first component assembly,	The first record is either an assembly manifest, or a whole assembly; the elements are other assemblies that are referenced as external in the first record; the first component assembly is a .NET application domain.
(1) at least one of said elements including at least some executable programming,	Assembly contains executable programming.
(2) at least one of said elements constituting a load module,	This is an external assembly referenced in the first record.
(i) said load module including executable programming and a header;	Assemblies include executable programming, and the assembly manifest and CLS type metadata constitute a header.
(ii) said header including an execution space identifier identifying at least one aspect of an execution space required for use and/or execution of the load module associated with said header;	This feature is provided for in the .NET architecture through numerous mechanisms, for example, by demands for ZoneID permissions.
(iii) said execution space identifier provides the capability for distinguishing between execution spaces providing a higher level of security and execution spaces providing a lower level of security;	SecurityZone or other evidence provides this capability.
(b) using said information to identify and locate said one or more elements;	Manifest and type metadata information section is used to identify and locate files, code elements, resource elements, individual classes and methods.
(c) accessing said located one or more elements;	Step carried out by the CLR or CCLR loader.
(d) securely assembling said one or more elements to form at least a portion of said first component assembly;	CLR or CCLR carries out this step, including checking the integrity of the load module, checking the load module's permissions, placing the load module contents into an application domain, isolating it from malicious or badly behaved code, and from code that does not have the permission to call it.
(e) executing at least some of said executable programming; and	Step carried out by the CLR/CCLR and the CLR/CCLR host.

(f) checking said record for validity prior to performing said executing step.	The CLR/CCLR checks the authenticity and the integrity of the first .NET assembly.
9. A process as in claim 8 in which said execution space providing a higher level of security comprises a secure processing environment.	The CLR/CCLR constitutes a secure processing environment.
13. A process as in claim 8 further comprising:	
(a) comparing said execution space identifier against information identifying the execution space in which said executing step is to occur; and	In one example, the ZoneIdentityPermissionAttribute SecurityZone value demanded by control in the assembly manifest is compared against the SecurityZone attribute value corresponding to the calling method
(b) taking an action if said execution space identifier requires an execution space with a security level higher than that of the execution space in which said executing step is to occur.	CLR/CCLR will throw an exception and transfer control to an exception handler in the calling routine; or it will shut down the application if there is no such exception handler, if the permissions do not include the permissions required by the ZoneIdentityPermissionAttribute. The ZoneIdentityPermissions are hierarchical, unless customized.
14. A process as in claim 13 in which said action includes terminating said process prior to said executing step.	CLR/CCLR may terminate the process or transfer control to an exception handler that may itself terminate the process.

Exhibit B

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP. INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 5,917,912

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
8.	Products infringing include Windows Installer SDK, and products that include the Windows Installer technology.
A process comprising the following steps:	Scenario 1: use of Windows Installer packages (i.e. .MSI files) to create Windows Installer-enabled applications, such as Office 2000 and used of the WI service to install them. Scenario 2: software distribution technologies that use the Windows Installer OS service for installation, such as Internet Component Download and products like Office Web Components. Either scenario can be used by SMS, IntelliMirror and third party tools like InstallShield and WISE. NT or later operating systems (because they use the subsystem identifier) using cabinet files, .CAB, (because they have a manifest and INF and/or OSD files), and have been signed with a digital signature and will be authenticated by Authenticode or WinVerifyTrust API and contain at least one PE (portable executables)
(a) accessing a first record containing information directly or indirectly identifying one or more elements of a first component assembly,	<u>Scenario 1</u>: First record is the .MSI file that contains information on what goes in the assembly and how to install the assembly. <u>Scenario 2:</u> A. First record is the cabinet manifest (indirect instructions) B. Or, First record can be INF and/or OSD files (direct instructions)
(1) at least one of said elements including at least some executable programming,	Both scenarios: The PE (portable executable) in the cabinet file is the executable programming.
(2) at least one of said elements constituting a load module,	Both scenarios: PE is a load module:
(i) said load module including executable programming and a	Both scenarios: The PE has several headers.

Exhibit B

header;	
(ii) said header including an execution space identifier identifying at least one aspect of an execution space required for use and/or execution of the load module associated with said header;	Both scenarios: SUBSYTEM is a field in the PE Optional Header that is an execution space
(iii) said execution space identifier provides the capability for distinguishing between execution spaces providing a higher level of security and execution spaces providing a lower level of security;	Both scenarios: SUBSYSTEM distinguishes between programs that can run in kernel mode and those that can run in user mode. This is a key security concept of process separation that was introduced with Windows NT. The Subsystem field in the PE header is used by the system to indicate whether the executable will run within Ring 3 (user mode) or use Ring 0 (native or kernel mode). Anything running in Ring 3 is limited to its own processing space. Executables running in Ring 0 can reach out to other spaces and have security measure built around them.
(b) using said information to identify and locate said one or more elements;	Scenario 1: the MSI file identifies and locates the elements Scenario 2: .CAB manifest is used to identify Physical location OSD and/or INF is used to identify Logical location
(c) accessing said located one or more elements;	Scenario 1: Using the MSI file Scenario 2: Using INF and/or OSD in cabinet file.
(d) securely assembling said one or more elements to form at least a portion of said first component assembly;	Both scenarios: Using the Window Installer OS service with various properties and flags on the settings for higher protection. Windows Installer has numerous flags that the developer can set to indicate how the assembly will be installed, in what privilege level, with how much user interface, and how much ability the user has to watch or change what is occurring. These controls have been strengthened with each release of Windows Installer. Windows Installer 1.1 and later has the ability to limit the users capabilities during the installation. In a Windows 2000

environment and later, using the Group Policy-based Change and Configuration Management, the administrator has the most control

Fields that can be set by the developer or administrator to control what users can do include the following:

Transformssecure can be set to a value of 1 to inform the installer that transforms are to be cached locally on the user's computer in a location the user does not have write access. (Transforms create custom installations from a basic generic installation, for example to make the Finance versions different from the Marketing version or English versions different from Japanese versions.)

AllowLockdownBrowse and *DisableBrowse* can prevent users from browsing to the sources.

SourceList can be used to specify the only allowable source to be used for the installation of a given component.

Environment can be used to specify whether the installation can be done while the user is logged on or only when no user is logged on.

Security Summary Property conveys whether a package can be opened as read-only or with no restriction.

Privileged Property is used by developers of installer packages to make the installation conditional upon system policy, the user being an administrator, or assignment by an administrator.

Restricted Public Properties can be set as variables for an installation. "For managed installations, the package author may need to limit which public properties are passed to the server side and can be changed by a user that is not a system administrator. Some are commonly necessary to maintain a secure environment when the installation requires the installer use elevated privileges. "

SecureCustomProperties can be created by the author of an installation package to add controls beyond the default list.

MsiSetInternalUI specifies the level of user interface from none to full.

A *Sequence Table* can be used to specify the required order of execution for the installation process. There are three modes, one of which is the *Administrative Installation* that is used by the network administrator to assign and install applications.

InstallServicesAction registers a service for the system and it can only be used if the user is

	an administrator or has elevated privileges with permission to install services or that the application is part of a managed installation. *DisableMedia* system policy disables media sources and disables browsing to media sources. It can be used with *DisableBrowse* to secure installations version 1.1 that doesn't have some of the other capabilities. *AlwaysInstallElevated* can be set per user or per machine and is used to install managed applications with elevated privileges. AllowLockdownBrowse, AllowLockdownMedia and AllowLockdownPatch set these capabilities so they can only be performed by an administrator during an elevated installation. [See article "HowTo: Configure Windows Installer for Maximum Security (Q247528). Windows XP Professional and .NET have the additional capability to set *Software Restriction Policies* and have these used by Windows Installer. In addition, most of the software distribution technologies that use Windows Installer also add a layer of their own controls. For example, SMS 2.0 enables the administrators to control the installation is optional or required and whether the user can affect the installation contents/features at all.
(e) executing at least some of said executable programming; and	Both scenarios: Part of executable is called during installation in order to do self-registration or perform custom actions. The overall executable is used at runtime.
(f) checking said record for validity prior to performing said executing step.	Scenario 1: Sign the overall package and the cabinet files. Scenario 2: The cabinet file is signed. For IE with the default security level or higher, the digital signature is verified by Authenticode or a similar utility before the component is allowed to be assembled.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 5,917,912

35.	Products infringing include all products that host the Microsoft .NET Common Language Runtime or Compact Common Language Runtime.
A process comprising the following steps:	
(a) at a first processing environment receiving a first record from a second processing environment remote from said first processing environment;	Computer running the Microsoft CLR/CCLR receives, for example, a shared assembly header or a complete shared assembly from another computer, for example a server.
(1) said first record being received in a secure container;	The shared assembly is cryptographically hashed and signed.
(2) said first record containing identification information directly or indirectly identifying one or more elements of a first component assembly;	The first record is either an assembly manifest, or a whole assembly; the elements are other assemblies that are referenced as external in the first record; the first component assembly is a .NET application domain.
(i) at least one of said elements including at least some executable programming;	Assembly contains executable programming.
(ii) said component assembly allowing access to or use of specified information;	The specified information can include any kind of data file, stream, log, environment variables, etc.
(3) said secure container also including a first of said elements;	The shared assembly includes at least some executable programming.
(b) accessing said first record	CLR/CCLR accesses the assembly or assembly header.
(c) using said identification information to identify and locate said one or more elements;	Manifest and type metadata information section is used to identify and locate files, code elements, resource elements, individual classes and methods.
(1) said locating step including locating a second of said elements at a third processing environment located remotely from said first processing environment and said second processing environment;	Met by a multifile assembly, with files distributed across a network, or by the second element constituting another referenced assembly located elsewhere; the CLR/CCLR uses probing to locate and access the file.
(d) accessing said located one or more elements;	Step carried out by the CLR/CCLR loader.
(1) said element accessing step including retrieving said second element from said third processing environment;	Step carried out by the CLR/CCLR loader.
(e) securely assembling said one or more elements to form at least a portion of said first component assembly specified by said first record; and	CLR/CCLR carries out this step, including checking the integrity of the load module, checking the load module's permissions, placing the load module contents into an application domain, isolating it from malicious or badly behaved code; and from code that

	does not have the permission to call it.
(f) executing at least some of said executable programming.	Step carried out by the CLR/CCLR.
(1) said executing step taking place at said first processing environment.	CLR/CCLR is operating in the first processing environment specified above.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,920,861

34.	Product Infringing: Microsoft Operating Systems that support device driver signature technology
A descriptive data structure embodied on a computer-readable medium or other logic device including the following elements:	
a representation of the format of data contained in a first rights management data structure	The driver package's INF is a data structure. The INF contains multiple types of sections, structured as hierarchy /"branches," that the Windows operating system or its Plug and Play and/or Set-up installation services "branch" through based on the operating system information and device for which a driver is to be installed. The installation services use the "branching" structure (format) to determine what files should be installed. The INF, further provides disk location information and file directory path information for the files identified as necessary as a result of the "branching" process. The driver package is a "rights management" data structure based on the fact that it is governed and based on the fact that it processes governed information. Rights Management as Governed Item A driver manufacturer can include rules governing the driver's installation and/or use in the driver's INF file. For example: Security entries specify an access control list for the driver. Driver developers can specify rules that determine behavior of the driver package based on the user's operating system version, including product type and suite and the device for which the driver is to be installed Rules specifying logging Local administrators can establish policy as to what action or notification should occur in the event that a driver being installed is not signed.

	The operating system installation services have a ranking criteria it follows when multiple drivers are available for a newly detected device. The criterion is used to determine the driver best suited for ensuring compatibility with the operating system and ensuring functionality of the device. Drivers have been certified to be compatible with specified operating system versions for their respective device classes. The catalog file protects the integrity of the driver. Microsoft distributes the Driver Protection List to prevent known bad deriver from being installed. <u>Processing Rights Managed Items</u> Certain drivers (SAP) have been explicitly certified to protect DRM content. <u>MSDN – DRM Overview</u> A DRM-compliant driver must prevent unauthorized copying while digital content is being played. In addition, the driver must disable all digital outputs that can transmit the content over a standard interface (such as S/PDIF) through which the decrypted content can be captured.
<u>said representation including:</u>	
element information contained within said first rights management data structure; and	The elements of a driver package include: A driver that is typically a dynamic-link library with the *.sys* filename extension. An INF file containing information that the system Setup components use to install support for the device. A driver catalog file containing the digital signature. One or more optional co-installers which are a Win32® DLL that assists in device installation NT-based operating systems. Other files, such as a device installation application, a device icon, and so forth. <u>XP DDK – INF Version Section</u> The LayoutFile entry specifies one or more additional system-supplied INF files that contain layout information on the source media required for installing the software

	described in this INF. All system-supplied INF files specify this entry. The CatalogFile entry specifies a catalog (*.cat*) file to be included on the distribution media of a device/driver.
organization information regarding the organization of said elements within said first rights management data structure; and	Within an INF is a hierarchy with the top being a list of manufacturers, and sub-lists of models and at the bottom a list of install information by model. For Windows XP and later versions of NT-based operating systems, entries in the **Manufacturer** section can be decorated to specify operating system versions. The specified versions indicate OS versions with which the specified INF *Models* sections will be used. If no versions are specified, Setup uses the specified *Models* section for all versions of all operating systems. INF's SourceDisksNames and SourceDisksFiles sections specify organization information. <u>XP DDK – Source Media for INFs</u> The methods you should use to specify source media for device files depend on whether your INFs ship separately from the operating system or are included with the operating system. INFs for drivers that are delivered separately from the operating system specify where the files are located using SourceDisksNames and SourceDisksFiles sections. If the files to support the device are included with the operating system, the INF must specify a **LayoutFile** entry in the Version section of the file. Such an entry specifies where the files reside on the operating system media. An INF that specifies a **LayoutFile** entry must not include **SourceDisksNames** and **SourceDisksFiles** sections. <u>**XP DDK – INF SourceDisksNames Section**</u> A **SourceDisksNames** section identifies the distribution disks or CD-ROM discs that contain the source files to be transferred to the target machine during installation. Relevant values of an entry in the INF include: *diskid* – Specifies a source disk. *disk-description* – Describes the contents

	and/or purpose of the disk identified by *diskid*. *tag-or-cab-file* – This optional value specifies the name of a tag file or cabinet file supplied on the distribution disk, either in the installation root or in the subdirectory specified by *path*, if any. *path*– This optional value specifies the path to the directory on the distribution disk containing source files. The *path* is relative to the installation root and is expressed as *\dirname1\dirname2*... and so forth. *flags* – For Windows XP and later, setting this to **0x10** forces Setup to use *cab-or-tag-file* as a cabinet file name, and to use *tag-file* as a tag file name. Otherwise, *flags* is for internal use only. *tag-file* – For Windows XP and later, if *flags* is set to **0x10**, this optional value specifies the name of a tag file supplied on the distribution medium, either in the installation root or in the subdirectory specified by *path*. The value should specify the file name and extension without path information.. <u>XP DDK – INF SourceDisksFiles Section</u> A **SourceDisksFiles** section names the source files used during installation, identifies the source disks (or CD-ROM discs) that contain those files, and provides the path to the subdirectories, if any, on the distribution disks containing individual files. Relevant values in an entry in the INF would include: *filename* -- Specifies the name of the file on the source disk. *diskid* -- Specifies the integer identifying the source disk that contains the file. This value and the initial *path* to the *subdir*(ectory), if any, containing the named file must be defined in a **SourceDisksNames** section of the same INF. *subdir* -- This optional value specifies the subdirectory (relative to the **SourceDisksNames** *path* specification, if any) on the source disk where the named file resides.
information relating to metadata, said metadata including:	
metadata rules used at least in part to govern at least one aspect of use and/or display of content stored within a rights management data structure,	The driver manufacture can specify rules in the INF that govern the installation and/or use of the driver. For example, security entries specify an access control list for the

Exhibit B

	driver. Driver developers can specify rules in an INF file that determines behavior of the driver package based on the user's operating system version, including product type and suite. Also, rules related to logging can be specified as mentioned in next claim element. **For Example – Access Control List Rules** **XP DDK – Tightening File-Open Security in a Device INF File** For Microsoft Windows 2000 and later, Microsoft tightened file-open security in the class installer INFs for certain device classes, including CDROM, DiskDrive, FDC, FloppyDisk, HDC, and SCSIAdapter. If you are unsure whether the class installer for your device has tightened security on file opens, you should tighten security by using the device's INF file to assign a value to the **DeviceCharacteristics** value name in the registry. Do this within an *add-registry-section*, which is specified using the INF AddReg directive. XP-DDK – INF AddReg Directive An INF can also contain one or more optional *add-registry-section*.**security** sections, each specifying a security descriptor that will be applied to all registry values described within a named *add-registry-section*. A **Security** entry specifies a security descriptor for the device. The *security-descriptor-string* is a string with tokens to indicate the DACL (D:) security component. A class-installer INF can specify a security descriptor for a device class. A device INF can specify a security descriptor for an individual device, overriding the security for the class. If the class and/or device INF specifies a *security-descriptor-string*, the PnP Manager propagates the descriptor to all the device objects for a device, including the FDO, filter DOs, and the PDO. For Example – Operating System Versioning Operating-System Versioning for Drivers

Exhibit B

	under Windows XP Setup selects the [*Models*] section to use based on the following rules: If the INF contains [*Models*] sections for several major or minor operating system version numbers, Setup uses the section with the highest version numbers that are not higher than the operating system version on which the installation is taking place. If the INF [*Models*] sections that match the operating system version also include product-type decorations, product suite decorations, or both, then Setup selects the section that most closely matches the running operating system.
said metadata rules including at least one rule specifying that information relating to at least one use or display of said content be recorded and/or reported.	The AddService directive can set up event-logging services for drivers. INF AddService Directive An AddService directive is used to control how (and when) the services of particular Windows 2000 or later device's drivers are loaded, any dependencies on other underlying legacy drivers or services, and so forth. Optionally, this directive sets up event-logging services by the devices/drivers as well. Relevant sections of the directive's entry include: *event-log-install-section* -Optionally references an INF-writer-defined section in which event-logging services for this device (or devices) are set up. *EventLogType* -- Optionally specifies one of System, Security, or Application. If omitted, this defaults to System, which is almost always the appropriate value for the installation of device drivers. For example, an INF would specify Security only if the to-be-installed driver provides its own security support. *EventName* -- Optionally specifies a name to use for the event log. If omitted, this defaults to the given *ServiceName*.

35. A descriptive data structure as in claim 34, in which:	
said first rights management data structure comprises a first secure container.	The driver package is secured through a catalog file that is signed by Microsoft's Windows Hardware Quality Lab and

	contains the hash of each file of the driver's package. The INF identifies the catalog file used to sign the driver package.
36. A descriptive data structure as in claim 35, in which:	
said first secure container comprises;	The first secure container is the driver package secured by a catalog file.
said content; and	The content is the driver and related files within the signed driver package.
rules at least in part governing at least one use of said content.	The rules are within the INF, which is part of the signed driver package.
37. A descriptive data structure as in claim 36, wherein the descriptive data structure is stored in said first secure container.	The INF is stored within the signed driver package.
44. A descriptive data structure as in claim 34, further including:	
a representation of the format of data contained in a second rights management data structure,	The manufacture and models sections in the INF Version section are provided for the possibility of a single INF representing the format for multiple drivers. Operating system version "decorating" relating the architecture, major and minor operating systems versions, product and suit information all relate to the target environment and is used to identify the files necessary for the target environment. An INF file, such as in the case of operating system targeting, can be used for more than one driver package since it can contain more than one catalog file. Further an INF can address the drives necessary for a multi-functional device.
said second rights management data structure differing in at least one respect from said first rights management data structure.	The files of the second data structure would vary from the files on the first data structure.
45. A descriptive data structure as in claim 44, in which:	
said information regarding elements contained within said first rights management data structure includes information relating to the location of at least one such element.	INF specify where the driver files are located using the SourceDiskNames and SourceDiskFiles sections.
46. A descriptive data structure as in claim 44, further including:	
a first target data block including information relating to a first target	Operating system version "decorating" relating the architecture, major and minor

Exhibit B

environment in which the descriptive data structure may be used.	operating systems versions, product and suit information all relate to the first target environment.

47. A descriptive data structure as in claim 46, further including:	
a second target data block including information relating to a second target environment in which the descriptive data structure may be used,	Operating system version decorating will cover multiple operating systems.
said second target environment differing in at least one respect from said first target environment.	This is the reason for version decorating.

48. A descriptive data structure as in claim 46, further including:	
a source message field containing information at least in part identifying the source for the descriptive data structure.	The provider entry in the version section of the INF identifies the provider of the INF file. Also, the INF contains a manufacture section.

Exhibit B

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 5,920,861

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
58.	Product Infringing: Microsoft Reader SDK and Microsoft Digital Asset Server.
A method of creating a first secure container, said method including the following steps:	Method is carried out by Microsoft's Digital Asset Server and Microsoft's .Litgen tools
(a) accessing a descriptive data structure, said descriptive data structure including or addressing	.opf file describing the file structure of a protected e-book including metadata, manifest, and "spine" information
(1) organization information at least in part describing a required or desired organization of a content section of said first secure container, and	Organization information regarding organization of the ebook and the inscription as specified in the manifest and spine information in the .opf file
(2) metadata information at least in part specifying at least one step required or desired in creation of said first secure container,	Metadata constitutes rules specifying the degree of security to use and/or XrML rules
(b) using said descriptive data structure to organize said first secure container contents	e-book packaging carried out by Microsoft Litgen tool
(c) using said metadata information to at least in part determine specific information required to be included in said first secure container contents; and	Step performed by Digital Asset Server; example of specific information is owner/purchaser information required in the inscription process
(d) generating or identifying at least one rule designed to control at least one aspect of access to or use of at least a portion of said first secure container contents.	Analyzing the metadata and finally packaging the e-book using a particular security level specified through the metadata
71. A method as in claim 58, in which:	
(a) said specific information required to be included includes information at least in part identifying at least one owner or creator of at least a portion of said first secure container contents.	Owner purchaser information required in the inscription process; XrML rule requiring display of copyright notice

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,920,861

58.	Product Infringing: All products that host the Microsoft Common Language Runtime or Compact Common Language Runtime.
A method of creating a first secure container, said method including the following steps;	Method is practiced by a user using the Common Language Runtime (CLR) or Compact Common Language Runtime (CCLR) to create a dynamic shared assembly or .NET Framework SDK to create a shared assembly
(a) accessing a descriptive data structure, said descriptive data structure including or addressing	.NET framework Assembly class and/or AssemblyBuilder class and/or AssemblyInfo file
(1) organization information at least in part describing a required or desired organization of a content section of said first secure container, and	This information is specified in the classes named above and in the AssemblyInfo file.
(2) metadata information at least in part specifying at least one step required or desired in creation of said first secure container;	This information is addressed in the classes and the AssemblyInfo file, e.g., for a shared assembly metadata will be specified that the assembly is to be signed using specified key
(b) using said descriptive data structure to organize said first secure container contents;	This step is carried out by applications and tools using the classes and assembly info file, including CLR (or CCLR) and .NET Framework SDK
(c) using said metadata information to at least in part determine specific information required to be included in said first secure container contents; and	This step is carried out by applications and tools using the assembly info file and classes that specify the metadata required in the target assembly.
(d) generating or identifying at least one rule designed to control at least one aspect of access to or use of at least a portion of said first secure container contents.	User may specify rules, as specified in the .NET Framework SDK, to be placed in the assembly manifest including such rules requiring that all code be managed (CLR or CCLR compliant), "Code Access Security" permissions be supplied for use of code supplied in the assembly, etc
64. A method as in claim 58, in which:	
(a) said creation of said first secure container occurs at a first data processing arrangement located at a first site;	Can be a server, PC or workstation running CLR (or CCLR) to create a dynamic shared assembly or .NET Framework SDK to create a shared assembly)
(b) said first data processing arrangement including a communications port; and	Included in virtually any computer
(c) said method further includes:	
(1) prior to said step of accessing said descriptive data structure, said	Download of the assemblyinfo file and/or a file containing a class calling the

Exhibit B

first data processing arrangement receiving said descriptive data structure from a second data processing arrangement located at a second site.	DefineDynamicAssembly methods or download of SDK containing assemblybuilder class from a second site
(d) said receipt occurring through said first data processing arrangement communications port.	Communications port is normally used for downloading
67. A method as in claim 64, further comprising:	
at said first processing site, receiving said metadata through said communications port.	Download of the AssemblyInfo file and/or a file containing a class calling the DefineDynamicAssembly methods or download of SDK containing assemblybuilder class from a second site
68. A method as in claim 67, in which,	
(a) said metadata is received separately from said descriptive data structure.	Method practiced when metadata names are addressed by the assembly class and a template for the AssemblyInfo file, and values corresponding to those names are received through a user interface such as provided by Microsoft Visual Studio or are provided from a separate file
71. A method as in claim 58, in which:	
(a) said specific information required to be included includes information at least in part identifying at least one owner or creator of at least a portion of said first secure container contents.	The Assembly class definition includes attributes for company name and trademark information, and these may be required attributes specified in the AssemblyInfo file
72. A method as in claim 58, in which:	
(a) said specific information required to be included includes a copyright notice.	The Assembly class definition includes an attribute for copyright field that may be required by the AssemblyInfo file

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,920,861

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
58.	Product Infringing: Microsoft .NET Framework, Visual Studio .NET, and tools that include the Assembly Generator tool AL.exe.
A method of creating a first secure container, said method including the following steps;	The Assembly Generation tool generates a portable execution file with an assembly manifest from one or more files that are either Microsoft intermediate language (MSIL) modules or resource files. When using the tool's signing option, the assembly becomes a *secure container*.
(a) accessing a descriptive data structure, said descriptive data structure including or addressing	The *descriptive data structure* is the text file used as input by the Assembly Generation tool.
(1) organization information at least in part describing a required or desired organization of a content section of said first secure container, and	The DDS specifies the *link* and or *embed* directives to indicate which source files should be included in the assembly, how the included resource will be tagged, and if the resource will be private. Private resources are not visible to other assemblies. These tags are used to organize the assembly into *named* sections. Private attributes are used to organize the assembly into both public and *private* sections. (Public sections are the default.)
(2) metadata information at least in part specifying at least one step required or desired in creation of said first secure container;	The text file can contain "options" relating to how the assembly should be built and additional information that should be included. *Main* – Specifies the method to use as an entry point when converting a module to an executable file. *Algid* – Specifies an algorithm to hash all files. *Comp* – Specifies string for the Company field. *Conf* – Specifies string for Configuration field *Copy* – Specifies string for Copyright field. *Culture* – Specifies the culture string to associate with the assembly. *Delay* – Variation of this option specifies whether the assembly will be

	fully or partially signed and whether the public key is placed in the assembly. *Description* – Specifies the description field. *Evidence* – Embeds file in the assembly with the resource name Security.Evidence. *Fileversion* – Specifies the file version of the assembly. *Flags* – Specifies flags for such things as the assembly is side-by-side compatible, assembly cannot execute with other versions if either they are executing in the same application domain, process or computer. *Keyf* – Specifies a file that contains a key or key pair to sign an assembly. *Keyn* – Specifies the container that holds a key pair. *Product* – Specifies string for Product field. *Productv* – Specifies string for Product Version. *Template* – Specifies the assembly fro which to inherit all assembly metadata. *Title* – Specifies string for Title field. *Trade* – Specifics string for Trademark field. *V* – Specifies version information.
(b) using said descriptive data structure to organize said first secure container contents	The following directives are used to specify which files are to be compiled into the assembly, how they will be tagged, and whether or not they will be visible to other assemblies, AKA private: *Embed[*name, *private]* – copies the content of the file into the assembly and applies an optional name tag, and optional private attribute. *Link[*name, *private]* – file becomes part of the assembly via a link and applies an optional name tag, and optional private attribute.
(c) using said metadata information to at least in part determine specific information required to be included in said first secure container contents; and	The following are some of the "options" address what information should be included in the secure container: *Main* – Specifies the method to use as an entry point when converting a module to an executable file. *Comp* – Specifies string for the Company field. *Conf* – Specifies string for Configuration field *Copy* – Specifies string for Copyright

Exhibit B

	field. *Culture* – Specifies the culture string to associate with the assembly. *Description* – Specifies the description field. *Evidence* – Embeds file in the assembly with the resource name Security.Evidence. *Fileversion* – Specifies the file version of the assembly. *Flags* – Specifies flags for such things as the assembly is side-by-side compatible, assembly cannot execute with other versions if either they are executing in the same application domain, process or computer. *Keyf* – Specifies a file that contains a key or key pair to sign an assembly. *Keyn* – Specifies the container that holds a key pair. *Product* – Specifies string for Product field. *Productv* – Specifies string for Product Version. *Template* – Specifies the assembly fro which to inherit all assembly metadata. *Title* – Specifies string for Title field. *Trade* – Specifics string for Trademark field. *V* – Specifies version information.
(d) generating or identifying at least one rule designed to control at least one aspect of access to or use of at least a portion of said first secure container contents.	User may specify rules, as specified in the .NET Framework SDK, to be placed in the assembly manifest including such rules requiring that all code be managed (CLR compliant), "Code Access Security" permissions be supplied for use of code supplied in the assembly, etc.
71. A method as in claim 58, in which:	
(a) said specific information required to be included includes information at least in part identifying at least one owner or creator of at least a portion of said first secure container contents.	The following "options" specifies owner and creator information: *Comp* – Specifies string for the Company field. *Copy* – Specifies string for Copyright field. *Trade* – Specifics string for Trademark field.
72. A method as in claim 58, in which:	
(a) said specific information required to be included includes a copyright notice.	The copy "option" specifies the string for the for the Copyright field.

Exhibit B

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 5,982,891

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
1.	Products infringing: All products that include the Common Language Runtime or Compact Common Language Runtime or Common Language Infrastructure.
A method for using at least one resource processed in a secure operating environment at a first appliance, said method comprising:	Resource may constitute a Microsoft Windows process or hardware element; secure operating environment is Microsoft Common Language Runtime ("CLR") environment, Common Language Infrastructure ("CLI") or Compact CLR ("CCLR"); first appliance is computer running CLR, CLI or Compact CLR. Two infringing scenarios are set forth herein: (1) For CLR, an administrator, using the .NET framework caspol.exe tool remotely configures security policy in a .NET configuration file for a machine, enterprise, user, or application and that security policy interacts with rules or evidence declared in a shared assembly provided by another entity ("1st scenario"); and (2) for CLR, CLI and CCLR two assemblies are delivered to an appliance; the first assembly has a rule that demands permissions from a caller in the second assembly, and the second assembly includes a control that asserts such permissions or provides evidence that convinces the runtime that it has such permissions. ("2nd scenario"). In each scenario Microsoft .NET "Code Access Security" framework or "Role Based Security" framework is used.
(a) securely receiving a first entity's control at said first appliance, said first entity being located remotely from said operating environment and said first appliance;	1st scenario: first entity is the administrator, and the policy that constitutes this entity's control is securely received at the first appliance through a session established between the administrator's computer and the first appliance, requiring security credentials such as the administrator's login and password or other secure session means. 2nd scenario: first entity is creator or distributor of the first assembly, assembly manifest includes a control demanding or refusing or otherwise asserting a security action on permissions from a caller; first assembly is integrity-checked.
(b) securely receiving a second entity's control at said first appliance, said second entity being located remotely from said operating environment and said first appliance, said second entity being different from said first	Second entity's control is contained in shared assembly manifest (and therefore integrity protected) that provides evidence for obtaining permissions, or asserts permissions; assembly creator/distributor is located remotely and is

entity; and	not the administrator (1st scenario) or creator/distributor of the first container (2nd scenario);
(c) securely processing a data item at said first appliance, using at least one resource, including securely applying, at said first appliance through use of said at least one resource said first entity's control and said second entity's control to govern use of said data item.	Secure processing is carried out by CLR, CLI or CCLR, Data item constitutes an executable code element, an interface controlled by such an executable, a data collection or stream (such as media file or stream or text file) or an environment variable. CLR, CLI or CCLR securely processes the rules, which will in both scenarios govern access to methods and data from the first assembly. The resource named in the claim is, e.g., a Windows process that is established by the runtime or hardware element on the computer.
51. A method as in claim 1 wherein at least said secure processing step is performed at an end user electronic appliance.	Consumer computer or appliance running Microsoft CLR, CLI or CCLR).
58. A method as in claim 1 wherein the step of securely receiving a first entity's control comprises securely receiving said first entity's control from a remote location over a telecommunications link, and the step of securely receiving said second entity's control comprises securely receiving said second entity's control from the same or different remote location over the same or different telecommunications link.	1st scenario 1: link is LAN or WAN; 2nd scenario: link is any telecommunications link, including the internet.
65. A method as in claim 1 wherein the processing step includes processing said first and second controls within the same secure processing environment.	Secure processing environment is CLR, CLI or CCLR running on user's computer or appliance.
71. A method as in claim 1 further including the step of securely combining said first entity's control and said second entity's control to provide a combined control arrangement.	In scenario 2, arrangement consists of the stack frame, and the corresponding array of permission grants for assemblies on the stack, and the permission demanded by the first assembly. Secure combining performed by the CLR, CLI or CCLR.
76. A method as in claim 1 wherein said two securely receiving steps are independently performed at different times.	Steps are performed at different times in both scenarios.
84. A method as in claim 1 wherein at least one of the first entity's control and the second entity's control comprises at least one executable component and at least one data component.	In both scenarios the second entity supplies an assembly with a demand procedure executed by the CLR, CLI or CCLR. The data component is a specific attribute value referenced by the assembly.
89. A method as in claim 1 wherein said first appliance includes a protected processing environment, and wherein:	Microsoft Common Language Runtime (CLR), Common Language Infrastructure (CLI), or Compact Common Language Runtime (CCLR) environment.
(a) said method further comprises a step of receiving, at said first appliance, said data item	Typically occurs in both scenarios.

separately and at a different time from said receiving said first entity's control ; and	
(b) said securely processing step is performed at least in part in said protected processing environment	Protected processing environment is the CLR, CLI or CCLR.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 5,982,891

22.	Infringing products include Office 2003 and included applications, and Server 2003, including Microsoft hosted RMS Service using Passport
A method of securely controlling use by a third party of at least one protected operation with respect to a data item comprising:	A user (third party) accesses an IRM-protected data item governed by IRM controls under two or more RMS servers. For example, the data item may be a IRM-protected document. The IRM controls may be associated with the data item directly or via a IRM-protected container holding the IRM-protected data item, such as an IRM-protected email with the IRM-protected document attached.
(a) supplying at least a first control from a first party to said third party;	The user acquires a first use license from a first RMS server (first party) enabling access to, the IRM-protected data item under the IRM rules associated with the first RMS server. For example: (1) the first use license from the first RMS server permits the user to access a IRM-protected document contained within or attached to an IRM-protected email; or (2) the first use license from the first RMS server applies a first set of IRM rules to an IRM-protected document.
(b) supplying, to said third party, at least a second control from a second party different from said first party;	The user acquires a second use license from a second RMS server (second party) enabling access to the IRM-protected data item under the IRM rules associated with the second RMS server. For example: (1) in addition to the user being given access to an IRM-protected email based on a first use license, a second RMS server provides a second use license enabling access to the IRM-protected document attached thereto; or (2) the second use license from the second RMS server applies a second set of IRM rules to the IRM-protected document.
(c) securely combining at said third party's location, said first and second controls to form a control arrangement;	The first and second use licenses are combined to form a control arrangement that governs access to the IRM-protected data item.
(d) securely requiring use of said control arrangement in order to perform at least one protected operation using said data item; and	The combined first and second use licenses govern access to the IRM-protected data item.
(e) securely performing said at least one protected operation on behalf of said third party with respect to said data item by at least in part employing said control arrangement	The user performs a protected operation (e.g., read, print, edit) on the IRM-protected data item. The combined first and second use licenses are employed to permit the protected operation.

23. A method as in claim 22 wherein said data item is protected.	The data item is encrypted and protected by IRM.
39. A method as in claim 22 further including securely and persistently associating at least one of: (a) said first control, (b) said second control, and (c) said control arrangement, with said data item.	The first and/or second use license are securely and persistently associated with the IRM-protected data item.
53. A method as in claim 22 wherein at least two of the recited steps are performed at an end user electronic appliance.	Steps performed at a user's computer or appliance.
60. A method as in claim 22 wherein step (a) comprises supplying said first control from at least one remote location over a telecommunications link, and step (b) comprises supplying said second control from the same or different remote location over the same or different telecommunications link	The first and second use licenses are received over a telecommunications link such as a networking or modem/serial interface.
67. A method as in claim 22 wherein at least step (c) is performed within the same secure processing environment at said third party's location.	Steps are performed at user's computer or appliance.
91. A method as in claim 22 wherein:	
(a) said method further comprises supplying said data item to said third party separately and at a different time from supplying of said first control to said third party; and	The first use license (first control) is received at the time that the user accesses the data item, which occurs separately and at a different time from receipt of the IRM-protected data item itself.
(b) said securely performing step comprises performing said protected operation at least in part in a protected processing environment.	The protected operations require decryption of the protected content, which is done inside the RM lockbox. The RM lockbox is protected by mechanisms such as obfuscation, anti-debugging, and tamper resistance.

Exhibit B

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,982,891

26.	Products infringing: Visual Studio.NET, .NET Framework SDK, and all products that include the Common Language Runtime or Compact Common Language Runtime or Common Language Infrastructure.
A secure method for combining data items into a composite data item comprising:	
(a) securely providing, from a first location to a second location, a first data item having at least a first control associated therewith;	A first signed and licensed .NET component, .NET assembly, managed control and/or Web control (component) is the first data item. The first .NET component developer (first location) provides the application assembly developer (second location) the first component. The first control is the set of declarative statements comprising the LicenseProviderAttribute (alternately referred to as license controls).
(b) securely providing, from a third location to said second location, a second data item having at least a second control associated therewith;	A second signed and licensed component is the second data item. The second component developer (third location) provides the application assembly developer (second location) the second component. The second control is the set of declarative statements comprising the LicenseProviderAttribute.
(c) forming, at said second location, a composite of said first and second data items;	The application assembly developer will include at least the two components into its assembly.
(d) securely combining, at said second location, said first and second controls to form a control arrangement; and	At the second location, the application assembly developer uses the .NET runtime that includes the LicenseManager. Whenever a component is instantiated (here, an instance of the first licensed component), the license manager accesses the proper validation mechanism for the component. The license controls (first control) for the runtime license (derived from the design-time license) are bound into the header of the .NET application assembly, along with the second control for the second component. Visual Studio.NET securely handles the creation of runtime license controls. Runtime licenses are embedded into (and bound to) the executing application assembly. The license control attribute

	included in the first component is customized in the second location to express and require the runtime license. In a more advanced scenario, the License Complier tool can be used to create a ".licenses file" containing licenses for multiple components, including runtime licenses for components and classes created by the license provider. This licenses file is embedded into the assembly. The third control set comprises the runtime license controls for the first and second components (that had been bound to the assembly), the declarative controls provided by the application assembly developer, and any runtime licenses for other components included by the developer in application assembly. The controls are typically integrated into the header of the .NET application assembly calling the first licensed component.
(e) performing at least one operation on said composite of said first and second data items based at least in part on said control arrangement.	The proper execution of the application will require that the assembly have run time licenses for the two components.
27. A method as in claim 26 wherein said combining step includes preserving each of said first and second controls in said composite set.	The set of declarative statements comprising the LicenseProviderAttribute of both the first and second components are included in the application assembly.
28. A method as in claim 26 wherein said performing step comprises governing the operation on said composite of said first and second data items in accordance with said first control and said second control.	The application will require the first and second controls to operate properly when it calls the first and second data items, respectively.
29. A method as in claim 26 wherein said providing step includes ensuring the integrity of said association between said first controls and said first data item is maintained during at least one of transmission, storage and processing of said first data item.	Signing the component that has embedded within it the license control ensures the integrity of the association of the control and data item.
31. A method as in claim 26 wherein said providing step comprises codelivering said first data item and said first control.	The component includes the license control and therefore they are codelivered.
40. A method as in claim 26 further including the step of securely ensuring that at least one of (a) said first control, (b) said second control, and (c) said control arrangement, is persistently associated with	Each component includes the license control. Signing the component that has embedded within it the license control ensures the persistence of the association of the control and data item.

Exhibit B

at least one of said first and second data items.	
54. A method as in claim 26 wherein at least one of steps (c), (d) and (e) is performed at an end user electronic appliance.	At least step (e) is typically performed at an end-user electronic appliance.
61. A method as in claim 26 wherein step (a) comprises providing said first data item from at least one remote location over a telecommunications link, and step (b) comprises providing said second data item from the same or different remote location over the same or different telecommunications link.	Microsoft maintains Web sites where a developer can get components over the Web. These sites include references whereby a developer may obtain components through their Web connection. One such site is Internet Explorer Web Control Gallery at ie.components.microsoft.com/webcontrols
68. A method as in claim 26 wherein step (d) is performed within the same secure processing environment at said second location.	Typically, step (d) will be performed within the same secure processing environment.
79. A method as in claim 26 wherein steps (a) and (b) are performed at different times.	The application assembly developer will typically acquire components at different times.
86. A method as in claim 26 wherein at least one of the first and second controls comprises at least one executable component and at least one data component.	The component must include an executable and can include a data items as a EULA, readme file or help file.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP. INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 5,982,891

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
35	Infringing products include: Windows Media Player, Individualized DRM Clients and the Secure Audio Path (SAP) technology.
A method for using at least one resource processed by a secure operating environment, said method comprising:	
securely receiving a first load module provided by a first entity external to said operating environment	The Individualized DRM Client (first load module) is a signed security upgrade DLL. It is also bound to the hardware ID of the machine on which it runs. It is therefore securely delivered and integrity protected.
securely receiving a second load module provided by a second entity external to said operating environment, said second entity being different from said first entity; and	A SAP certified driver is also signed and carries with it a certificate that indicates its compliance with SAP criteria. If it is delivered to a PC it is secure in the sense that it is integrity protected. This driver would not come from the same entity as the Individualization DLL.
securely processing, using at least one resource, a data item associated with said first and second load modules, including securely applying said first and second load modules to manage use of said data item.	If a WM audio file targeted to the Individualized DRM client carries with it a requirement that SAP be supported to render the WMF contents, the content is processed for playing through a soundcard using the WMP and by applying the DRM client - which decrypts the content and negotiates with the DRM kernel processing of the content through a Secure Audio Path that includes the SAP-certified audio driver.
56. A method as in claim 35 wherein at least two of the recited steps are performed at an end user electronic appliance.	All steps occur at the user's PC that supports the WMP and DRM client and SAP.
63. A method as in claim 35 wherein said first load module receiving step comprises securely receiving said first load module from at least one remote location over at least one telecommunications link, and said second load module receiving step comprises securely receiving said second load module from the same or different remote location over the same or different telecommunications link.	The Driver and DRM client are received from distinct locations and may be delivered securely over the Internet. They are delivered securely in that each is integrity protected.
70. A method as in claim 35 wherein said securely processing step comprises securely executing said first and second	Both load modules are executed on the PC within the WMP/DRM Client/SAP environment.

Exhibit B

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
load modules within the same secure processing environment.	
74. A method as in claim 35 further including securely combining said first and second load modules to provide a combined executable.	Since both the DRM client and the driver are DLLs in the same audio rendering chain, they exist as an execution environment.
81. A method as in claim 35 wherein said securely receiving steps are performed independently at different times.	The driver and Individualization DLL need not be received at the same time.
94. A method as in claim 35 wherein said secure operating environment includes a protected processing environment, and wherein: said method further comprises receiving a data item within said secure operating environment; said first load module receiving step is performed separately and at a time different from receiving said data item; and said securely processing step is performed at least in part in said protected processing environment.	The Windows Media Player together with the Individualized DRM Client and Secure Audio Path comprise a protected environment for processing protected media. The protected Windows Media Files are received after the load modules have been received and installed (licenses cannot be acquired until load modules are in place). The processing of the Windows Media File occurs in the protected environment.

Examples of SAP-certified drivers include - as indicated at http://www.microsoft.com/Windows/windowsmedia/WM7/DRM/FAQ.asp#Security7

- All VIA controllers with AC-97 codecs
- All ALI controllers with AC-97 codec
- Intel ICH controllers with AC-97 codecs
- Creative Labs SoundBlaster16/AWE32/AWE64/Vibra
- Yamaha OPL3
- Yamaha DS-1
- Cirrus Logic (Crystal) CS4280
- Cirrus Logic (Crystal) CS4614 / CS4624
- ESS Maestro 2E
- USB Audio
- Cirrus Logic (Crystal) CS4281

Exhibit B

- All SIS controllers with AC-97 codecs
- Ensoniq ES1370
- NeoMagic NM6
- Ensoniq ES1371/73 and CT5880
- SoundBlaster Live!
- Aureal 8810
- Aureal 8820
- Aureal 8830
- Conexant Riptide
- ESS Maestro
- ESS ISA parts
- NeoMagic NM5

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 5,982,891

36.	Product Infringing: Any product using Common Language Runtime (CLR), Common Language Infrastructure (CLI), or Compact Common Language Runtime (CCLR)
A secure operating environment system for managing at least one resource comprising:	Microsoft CLR, CLI or CCLR (operating environment system), managing any of the resources on a typical computer, including memory, files system, communications ports, storage devices, and higher level resources that may use any of these or combinations of them.
(a) a communications arrangement	Communications port and Microsoft Internet Protocol stack that may optionally use Secure Socket Layer protocol or IPSEC packet security protocol, supplied with Microsoft Windows.
(1) that securely receives a first control of a first entity external to said operating environment, and	Rule or evidence contained in the manifest of a shared assembly, distributed by a first entity that can be used by the CLR, CLI or CCLR to determine permissions that may be needed to cause operations on a data item or resource controlled by another entity; shared assembly is tamper-protected and may be received using secure SSL or IPSEC protocol.
(2) securely receives a second control of a second entity external to said operating environment, said second entity being different from said first entity; and	Rule specified in the manifest of a second shared (Tamper protected) assembly, that demands permissions of callers of its methods.
(b) a protected processing environment, operatively connected to said communications arrangement, that:	CLR, CLI or CCLR, connected to (e.g.) communications port
(1) [] securely processes, using at least one resource, a data item logically associated with said first and second controls, and	CLR, CLI or CCLR uses type safety mechanisms, access controls, integrity detection, and separation of domains. Data item may be any data item that is managed by the second assembly, which may be a member of such assembly, and whose state or value may be accessible through an interface to other assemblies, and which is referenced by the first assembly.
(2) [] securely applies said first and second controls to manage said resource for controlling use of said data item.	CLR, CLI or CCLR processes the demand for permissions from the second assembly, collects the evidence or processes the rule from the first assembly, and determines whether the first assembly has the permissions to use the resource to operate on the data item controlled by the second assembly.
57. A system as in claim 36 wherein said protected processing environment is part of an	Computer or electronic appliance running CLR, CLI or CCLR

Exhibit B

end user electronic appliance.	
64. A system as in claim 36 wherein said communications arrangement receives said first and second controls from at least one remote location over at least one telecommunications link.	Shared assemblies are designed to be received remotely, e.g., over the internet.
75. A system as in claim 36 wherein said protected processing environment combines said first and second controls to provide a combined control arrangement.	Arrangement consists of the stack frame and and the corresponding array of permission grants for assemblies on the stack, and the permission demanded by the second assembly.
82. A system as in claim 36 wherein said communications arrangement independently receives said first and second controls at different times	Assemblies, including controls, are designed for independent delivery.
88. A system as in claim 36 wherein at least one of the first control and second controls comprises at least one executable component and at least one data component.	The second entity supplies an assembly with a demand procedure (executed by the CLR, CLI or CCLR) that includes reference to a specific attribute value (the data component), and the protected processing environment executes the executable component (demand) in a manner that is at least in part responsive to the data component (execution is in response to the security action supplied in the data item).

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 5,982,891

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
36.	Infringing Product: My Services
A secure operating environment system for managing at least one resource comprising:	Secure operating environment is the secure server for any .NET My Services service (e.g. My Calendar, My Inbox)
a communications arrangement that securely receives	Secure server receives communications formatted using the SOAP-SEC, the security extension to SOAP that is used by My Service servers to receive controls.
a first control	The first control is a roleTemplate associated with the service. The roleTemplate identifies specific actions (e.g. read, replace) that can be performed against a certain scope (resource or set of resources).
of a first entity external to said operating environment,	The first entity is the administrator of the server database, or other entity with authority over its content that sets up the roleTemplates and scopes. That entity is independent from and located remotely from the secure server.
and securely receives a second control	A role element specified by a specific end user, which is securely received by the secure server using the SOAP-SEC protocol..
of a second entity external to said operating environment, said second entity being different from said first entity;	The end user is located remotely from the secure server.
and a protected processing environment, operatively connected to said communications arrangement, that:	The protected processing environment is the .NET security service (authorization system) operating within the server. The server uses the SOAP-SEC communication protocol to receive controls.
(a) securely processes, using at least one resource, a data item logically associated with said first and second controls, and	"Securely processes" is performing the requested operation on secure server running .NET. The system will perform the requested operation ensuring that the user has no access to information outside the

Exhibit B

	scope computed. The resource is the server software and/or hardware used to process the two controls and user data. The first control is the roleTemplate for the service. The second control is the role element for an individual user. The data item is the end user's stored content (e.g. calendar, email inbox, etc.).
(b) securely applies said first and second controls to manage said resource for controlling use of said data item.	The secure server determines the result scope (visible node set) for the operation that is computed from the role element and the roleTemplate. That result scope is used to manage the data item.
64. A system as in claim 36 wherein said communications arrangement receives said first and second controls from at least one remote location over at least one telecommunications link.	The remote location is the site where the user's or administrator's application is running. The telecommunication link can be the Internet, intranet, VPN or other similar channels.
75. A system as in claim 36 wherein said protected processing environment combines said first and second controls to provide a combined control arrangement.	The role scope incorporating the role element and the role Template.
82. A system as in claim 36 wherein said communications arrangement independently receives said first and second controls at different times.	Administrator and user controls will ordinarily be received at different times.
95. A secure operating environment system as in claim 36 wherein said communications arrangement also receives a data item separately and at a different time from at least one of said first control and said second control.	This is the normal case for .NET My Services. The user's content is normally stored and updated independently of the setting of scope elements, role elements and roleTemplates.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP. INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 6,157,721

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT Product Infringing: Windows CE for Automotive
1. A security method comprising:	WCEfA is Microsoft Windows CE for Automotive, sometimes also known by its former name, AutoPC 2.0. With WCEfA an OEM can assign their device to a class that only accepts certain kinds of software. The device can be set to accept 1) any software with the correct processor/version 2) only certified software or 3) only software from the OEM or Microsoft. These Security (or Trust) levels also control to which kernel APIs and middleware APIs the software has access. Background: "Microsoft Software Install Manager (SIM), a component of WCEfA, allows you to control what can be installed on your device platform. You can define your platform as being open, closed or restricted to new installations, and SIM will enforce these designations." (D,pg.1) "Anything can be installed on an open platform, as long as the applications are compiled for the appropriate processor. At the other extreme, no third-party software can be installed on a closed platform. Only certified applications can be installed on a restricted platform." (D, pg.1) "By restricting installations to compliant applications, the risk of installing and using incompatible or harmful software is greatly reduced, while still keeping the device open for robust, quality applications that enhance the user experience." (F, pg.1) WCEfA also has a Security Layer whose purpose is to "Create an abstraction layer of security surrounding ISV applications to limit and/or deny access to key Windows CE kernel API calls and WCEfA middleware APIs." I, pg. 1)
(a) digitally signing a first load module with a first digital signature designating the first load module for use by a first device class;	A *first load module* is a WCEfA software component in a signed .PE file. The *first device class* is a device that only allows software designated as "restricted" (or higher) to be installed. "Restricted" software is software that has been certified. With restricted software, the device also implements a Security Layer functionality that limits the kernel and WCEfA API calls that the software can make.

	"SIM Level: 1 = Restricted Description: Only properly certified CEI (WCEfA device installation) files can be installed on the device. Remote execution is restricted to executables with master key. Key: Logo certified CEI file required. CEI files or EXEs with master keys permitted." (F, pg. 1) "The kernel loader calls it each time a module is loaded by Windows CE. It returns one of the following values that determine the module's access to kernel resources: **Value** **Meaning** OEM_CERTIFY_TRUST (2) The module is trusted by the OEM to perform any operation. OEM_CERTIFY_RUN (1) The module is trusted by the OEM to run but is restricted from making certain function calls. OEM_CERTIFY_FALSE (0) The module is not allowed to run. " (H, pg. 1) *Digitally signing*: "Before the kernel loads a file, it uses the OEMCertifyModule function to verify that the file contains the proper signature." (N, pg.1) "Signfile.exe: This tool signs an executable with a supplied private key. You can use the following command parameters with this tool....-s AttribString, specifies an optional attribute string to be included in the signature. For example, you could add a string to indicate the trust level of the application." (O, Pg. 1) In the MSDN article <u>Verifying the Signature</u>, the sample code segment states "//the file has a valid signature //.we expect the trust level to be returned as signed data... //case 'R' : dwTrustLevel = OEM_CERTIFY_RUN" (N, pg.2) "The WCEfA Security Layer isolates installed applications from making unrestricted kernel and WCEfA API calls. This allows the OEM to assign one of three levels of security to applications and drivers installed in RAM when they are loaded into the system. The three levels are Trusted...,Restricted..., and Blocked...On the systems level, the WCEfA Security

	layer fits between ISV applications and isolates these software modules from having free access to all WinCE kernel calls and WCEfA middleware APIs." (I, pg. 1) The developer submits their application for certification. If it passes, then the .cei file (a form of cab file) receives a certification key from the certifier. The signed PE is within this .cei file.

(b) digitally signing a second load module with a second digital signature different from the first digital signature, the second digital signature designating the second load module for use by a second device class having at least one of tamper resistance and security level different from the at least one of tamper resistance and security level of the first device class;	A *second load module* is a WCEfA software component is a signed PE file. The *second device class* with a different tamper resistance or security level is a device that is "Closed", that is, it will not allow third party to software to be installed. A closed device only allows trusted software to run. The Security Layer setting of "Trusted" allows the Microsoft and OEM software full access to kernel and middleware APIs.
	
	In the MSDN article <u>Verifying the Signature</u>, the sample code segment states
	"//the file has a valid signature
	// we expect the trust level to be returned as signed data...
	//case 'T' : dwTrustLevel = OEM_CERTIFY_TRUST" (N, pg.2)
	
	"Signfile.exe: This tool signs an executable with a supplied private key. You can use the following command parameters with this tool....-s AttribString, specifies an optional attribute string to be included in the signature. For example, you could add a string to indicate the trust level of the application. (O. Pg. 1)
	
	"SIM Level: 2 = Closed
	Description: Platform is limited to software supplied directly by OEM or Microsoft. Third-party applications cannot be installed. ...
	Key: Master key required for any install or remote execution." (F, pg.1)
	
	Related to the Security Layer, the Trusted level "is most likely reserved for MS and OEM applications and drivers." (I, pg. 1)
	
	Whereas the .cei files for certified software have a certification key (sometimes call MS Logo key), the .cei files from Microsoft or the OEM have a master key attached. ""Master key required for any install or remote execution." (F, p.g1)

(c) distributing the first load module for use by at least one device in the first device class; and	*First load module* is the certified software from a third party that will be run as part of the "Restricted" *first device class*. "Once your application is complete, send the .cei file to

	the organization that is performing validation or certification for the OEM. They would validate it, then either reject or return a .cei that has been stamped with a certification key. You would then reproduce this .cei file on CD-ROM or a compact flash card and distribute." (D, p.g 5) "APCLoad compares the device SIM level against the .cei file certification key, and either allows the installation to proceed or prohibits it based on the outcome of this comparison." (D, pg. 2) "Security:. To achieve a high level of reliability, WCEfA is carefully designed to: - Control the installation of certified and tested software and drivers. - Limit the access of system services by installed module. - Monitor the proper execution of software..." (G, pg. 1)
(d) distributing the second load module for use by at least one device in the second device class.	The *second load module* is the certified software from the OEM or Microsoft that will be run as part of the "Closed" *second device class.* "You may need to change ROM components after your device ships, either to fix a problem, or to provide enhanced functionality. For this purpose, the OEM is given a CEIBuild that adds a master key to a .cei file. CEI files stamped with this master key can be installed on an open, closed or a restricted platform." (D, pg. 3) "Trusted: The application is registered as a completely trusted module and allowed full access to the kernel APIs and WCEfA APIs. This mode is mostly likely reserved for MS and OEM applications and drivers. Note that applications and drivers included in ROM are automatically given trusted status." (I, pg.1)

References:
[D] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnceauto/html/WinCAuto_SIM.asp
[F] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/apcguide/htm/ceibuildrev_8.asp
[G] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/apcguide/htm/securityrev.asp
[H] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/apcguide/htm/securityrev_7.asp
[I] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/apcguide/htm/reliabilityrev_3.asp
[N] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wcedsn40/htm/cgconVerifyingSignature.asp
[O] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wceoem/htm/os_secur_6.asp

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,157,721

5.	Product infringing: Windows Hardware Quality Lab certification services, and operating system products that support driver signature technology.
A software verifying method comprising:	Microsoft encourages manufacturers to have their device drivers tested and signed. For example, only signed drivers will ship "in-the-box." Also, Microsoft's driver ranking prefers signed drivers to unsigned drivers. Microsoft Web Page – Can't Find a Test Category for Your Driver? WHQL's long-term objective is to be able to digitally sign all drivers. Although we do not currently have test programs for certain driver types, such as specialized device drivers and software filter drivers, WHQL is investigating a long term solution to expand the categories of drivers tested under Windows 2000 and ultimately all Windows operating systems. We are already formulating a test program for anti-virus file system filters, and plan to address other file system filter drivers as soon as the initial program is in place.
(a) testing a load module	The driver will be tested for each version of the operating system it supports and against the device class specification that apply to the device's class. The driver package is a load module. A driver package contains one or more of the following files: A device setup information file (INF file) A driver catalog (.cat) file One or more optional co-installers Microsoft operates the Window Hardware Quality Lab, which tests drivers submitted by driver manufactures. The manufacturer can test their own driver using the Microsoft testing kit and submit the test results to WHQL when requesting a signature. Additionally, Microsoft or a testing facility working with Microsoft can perform the testing.
having at least one specification associated	The manufacturer-written INF file, which

therewith,	is part of the driver package, is a specification. Microsoft Windows drivers must have an INF file in order to be installed.
the specification describing one or more functions performed by the load module;	The INF Version section specifies its device class. One use of the device class is to identify the specific Windows compatibility specification that relate to the device class. These specifications will vary by device class in part because the function of each device can vary among class. The INF incorporates by reference the Microsoft supplied device class-specific specification by identifying its class in the INF. The INF can include operating system "decorating" to specify the operating system architecture, major and minor version, product and suite the driver is intended for and can further use this decorating to specify what operating systems for which it is not intended. Because the functionality of each of the operating systems may vary the driver must be tested for each applicable operating system. <u>Qualification Service Policy Guide – Hardware Category Policies</u> You must select the correct hardware category for your device. If you select the wrong hardware category for your device, your submission will fail. For example, if you have a storage/hard drive device, but you select storage/tape drive as your hardware category, your submission will fail. Windows XP HCT 10.0 Q & A – Windows XP Logos Q: Which "Designed for Windows XP" logos are available for my product? A: Devices and systems qualify for a "Designed for Windows" logo after passing testing with the appropriate WHQL test kit on all operating systems specified by the logo. "Designed for Windows" Logos for Device and System Programs lists which logos are available for each type of product.
(b) verifying that the load module satisfies the specification; and	The Microsoft WindowsXP Hardware Compatibility Test (HCT) kit version 10.0 includes the tests, test documentation, and

	submission processes that are required to participate in the Microsoft Windows Logo Program for Hardware for the Windows XP Professional operating system. To qualify to use the "Designed for Windows" logo for hardware, products must pass testing with the Microsoft Windows HCT kit. The HCT kits are organized by hardware type. As mentioned above, the manufacturer can test their own driver using the Microsoft testing kit and submit the test results to WHQL when requesting a signature. Additionally, Microsoft or a testing facility working with Microsoft can perform the testing.
(c) issuing at least one digital certificate attesting to the results of the verifying step.	When a driver package passes WHQL testing, WHQL generates a separate CAT file containing a hash of the driver binaries and other relevant information. WHQL then digitally signs the CAT file using Digital Signature cryptographic technology and sends it to the vendor. Driver signing does not change the driver binaries or the INF file submitted for testing. Microsoft uses digital signatures for device drivers to let users know that drivers are compatible with Microsoft Windows XP, Windows 2000, and Windows Me. A driver's digital signature indicates that the driver was tested with Windows for compatibility and has not been altered since testing.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,157,721

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
14.	Infringing products include Office 2003 and included applications, and Server 2003, including Microsoft hosted RMS Service using Passport
A first protected processing environment comprising:	A personal computer running Windows XP, Windows 2000, or Windows 2003
a first tamper resistant barrier having a first security level, and	The tamper resistant barrier is the Office 2003 IRM client environment and includes the signed digital certificate identifying the user. If the certificate is tampered with, or if certain, sensitive IRM processes or modules are debugged or tampered with, the system will cease to operate. The first security level is the "Security Level" which has been selected for a particular Office Application, e.g., Word.
at least one arrangement within the first tamper resistant barrier that prevents the first protected processing environment from executing the same load module accessed by a second protected processing environment having a second tamper resistant barrier with a second security level different from the first security level.	The arrangement that prevents a load module from running in one PPE and not in another is the type and characteristics of a particular Load Module (VBA program within a document or add-in); i.e., signed, script author, code capabilities, etc., and the "Security Level" settings.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 6,157,721

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
18.	Infringing products include Office 2003 and included applications, and Server 2003, including Microsoft hosted RMS Service using Passport
A method for protecting a first computing arrangement surrounded by a first tamper resistant barrier having a first security level, the method including:	The first computing arrangement with a tamper resistant barrier is the Office 2003 IRM client environment and includes the signed digital certificate identifying the user. If the certificate is tampered with, or if certain, sensitive IRM processes or modules are debugged or tampered with, the system will cease to operate. The computing arrangement is being protected from; for example, viruses and malicious code. The first security level is the "Security Level" which has been selected for a particular Office Application, e.g., Word.
preventing the first computing arrangement from using the same software module accessible by a second computing arrangement having a second tamper resistant barrier with a second security level different from the first security level.	The arrangement that prevents a load module from running in one computing arrangement and not in another is the type and characteristics of a particular software module (VBA program within a document or add-in); i.e., signed, script author, code capabilities, etc., and the "Security Level" settings.

Exhibit B

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 6,157,721

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
34.	Infringing products include Office 2003 and included applications, and Server 2003, including Microsoft hosted RMS Service using Passport
A protected processing environment comprising:	A personal computer running Windows XP, Windows 2000, or Windows 2003
a first tamper resistant barrier having a first security level,	The first tamper resistant barrier is the Office 2003 IRM client environment and includes the signed digital certificate identifying the user. If the certificate is tampered with, or if certain, sensitive IRM processes or modules are debugged or tampered with, the system will cease to operate. The first security level is the "Security Level" which has been selected for a particular Office Application, e.g., Word.
a first secure execution space, and	The secure execution space is process space allocated by the operating system for the Microsoft Office host application to run. This host application (e.g., Word) executes the VBA code within this process space. This execution space (application) is secure because the IRM environment takes steps to insure that it is "trusted", the application is signed, and the document which includes the VBA code is protected by IRM policy and then encrypted and signed.
at least one arrangement within the first tamper resistant barrier that prevents the first secure execution space from executing the same executable accessed by a second secure execution space having a second tamper resistant barrier with a second security level different from the first security level.	The arrangement that prevents a load module from running in one computing arrangement and not in another is the type and characteristics of a particular software module (VBA program within a document or add-in); i.e., signed, script author, code capabilities, etc., and the "Security Level" settings.

Exhibit B

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,157,721

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
34.	Product Infringing: Microsoft Common Language Runtime and ASP.NET
A protected processing environment comprising:	Microsoft Common Language Runtime and ASP.NET

a first tamper resistant barrier having a first security level,	TAMPER RESISTANT BARRIER
	The first tamper resistant barrier is the application domain in the CLR. The runtime hashes the contents of each file loaded into the application domain and compares it with the hash value in the manifest. If two hashes don't match, the assembly fails to load.[1]
	
	Also "*Code running in one application cannot directly access code or resources from another application. The common language runtime enforces this isolation by preventing direct calls between objects in different application domains. Objects that pass between domains are either copied or accessed by proxy.*"[2]
	
	SECURITY LEVELS
	
	The security levels of the application domain if different by setting the trust level assigned to an outside application using the "trust" element in the web.config for the ASP.NET application.
	Syntax-
	`<trust level="Full/High/Low/None" originUrl="url"/>`
	
	Example-
	`<trust level="High" originUrl=http://www.SomeOtherCompany.com/default.aspx />`
	
	[7]

a first secure execution space, and	The application domain is the execution space for a particular application.
at least one arrangement within the first tamper resistant barrier that prevents the first secure execution space from executing the same executable accessed by a second secure execution space having a second tamper resistant barrier with a second security level different from the first security level.	The second secure execution space is another application domain that has a different trust level for an outside application. If second app domain gives Full trust to the outside application; whereas the first one doesn't, the first app domain won't be able to execute the application that requires full trust permission.
	References: [1]

Exhibit B

	www.microsoft.com/germany/ms/msdnbiblio/dotnetrk/doc/assembly.doc [2] msdn.Microsoft.com/library/en-us/cpguide/html/cpconapplicationdomainsoverview.asp?frame=true [7] LaMacchia,etc, .NET Framework Security, Addision-Wesley, 2002

Exhibit B

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,157,721

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
34.	Product Infringing: Products containing Microsoft Common Language Runtime or Compact Common Language Runtime and products implementing the Common Language Infrastructure specification.
A protected processing environment comprising:	Microsoft Common Language Runtime and .NET Framework SDK:
a first tamper resistant barrier having a first security level,	TAMPER RESISTANT BARRIER The first tamper resistant barrier is the application domain in the CLR. The runtime hashes the contents of each file loaded into the application domain and compares it with the hash value in the manifest. If two hashes don't match, the assembly fails to load. [1] Also "*Code running in one application cannot directly access code or resources from another application. The common language runtime enforces this isolation by preventing direct calls between objects in different application domains. Objects that pass between domains are either copied or accessed by proxy.*"[2] SECURITY LEVELS Application domains have different security levels by setting security policy of the application domain programmatically. [3] "*It has different security based on code-based security model of .NET. Administrators and hosts use code-access security to decide what code can do, based on characteristics of the code itself, regardless of what user is executing the code. The code characteristics are called* evidence *and can include the Web site or zone from which the code was downloaded, or the digital signature of the vendor who published the code.*" "*When the security manager needs to determine the set of permissions that an assembly is granted by security policy, it starts with the enterprise policy level. Supplying the assembly evidence to this policy level will result in the set of permissions granted from that policy level. The security manager typically continues to collect the permission sets of the policy levels below the enterprise policy* [including the app domain] *in the same*

	fashion. These permission sets are then intersected to generate the policy system permission set for the assembly. All levels must allow a specific permission before it can make it into the granted permission set for the assembly." Example of granted permission sets from a policy – *Condition: All code, Permission Set: Nothing* *Condition: Zone: Internet, Permission Set: Internet Condition: URL: www.monash.edu.au, Permission Set: MonashPSet* *Condition: Strong Name: m-Commerce, Permission Set: m-CommercePSet [4]* Another difference in security levels can be whether the verification process is turned off or on, "*Managed code must be passed through a verification process before it can be run (unless the administrator has granted permission to skip the verification). The verification process determines whether the code can attempt to access invalid memory addresses or perform some other action that could cause the process in which it is running to fail to operate properly. Code that passes the verification test is said to be type-safe. The ability to verify code as type-safe enables the common language runtime to provide as great a level of isolation as the process boundary, at a much lower performance cost.*" [5]
a first secure execution space, and	The application domain is the execution space for a particular application.
at least one arrangement within the first tamper resistant barrier that prevents the first secure execution space from executing the same executable accessed by a second secure execution space having a second tamper resistant barrier with a second security level different from the first security level.	The second secure execution space is another application domain that has a different security policy than the first. If second app domain's security policy doesn't give any permission to code from internet zone, but first app domain does, then the code would run in first app domain and not in second.[6]
	References: [1] www.microsoft.com/germany/ms/msdnbiblio/dotnetrk/doc/assembly.doc [2] msdn.Microsoft.com/library/en-us/cpguide/html/cpconapplicationdomainsoverview.asp?frame=true

Exhibit B

[3] LaMacchia,etc, .NET Framework Security, Addision-Wesley, 2002, p.113
[4] Watkins, Demien, "An Overview of Security in the .NET Framework", from MSDN Library, January 2002
[5] same as [2]
[6] msdn.Microsoft.com/library/en-us/cpguide/html/cpconapplicationdomainlevelsecuritypolicy.asp?frame=true

Exhibit B

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 6,157,721

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
38.	Infringing products include Office 2003 and included applications, and Server 2003, including Microsoft hosted RMS Service using Passport
A method for protecting a first computing arrangement surrounded by a first tamper resistant barrier having a first security level, the method including:	The first computing arrangement surrounded by a tamper resistant barrier is the Office 2003 IRM client environment and includes the signed digital certificate identifying the user. If the certificate is tampered with, or if certain, sensitive IRM processes or modules are debugged or tampered with, the system will cease to operate. The first security level is the "Security Level" which has been selected for a particular Office Application, e.g., Word.
preventing the first computing arrangement from using the same software module accessed by a second computing arrangement having a second tamper resistant barrier with a second security level different from the first security level.	The computing arrangement that prevents a software module from running in one computing arrangement and not in another is the type and characteristics of the particular software module (VBA program within a document or add-in); i.e., signed, script author, code capabilities, etc., and the "Security Level" settings.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,185,683

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
2.	**Product Infringing: Windows Media Rights Manager and Windows Media Player**
A system including:	
(a) a first apparatus including,	Consumer's computer, as shown in WMRM SDK
(1) user controls,	Consumer's computer, as shown in WMRM SDK
(2) a communications port,	Consumer's computer, as shown in WMRM SDK
(3) a processor,	Consumer's computer, as shown in WMRM SDK
(4) a memory storing:	Consumer's computer, as shown in WMRM SDK
(i) a first secure container containing a governed item, the first secure container governed item being at least in part encrypted; the first secure container having been received from a second apparatus;	Secure container (packaged Windows Media file), received by consumer's computer from "Content provider" (WMRM SDK, Step 3), which contains encrypted governed item ("Encrypted content")
(ii) a first secure container rule at least in part governing an aspect of access to or use of said first secure container governed item, the first secure container rule [sic], the first secure container rule having been received from a third apparatus different from said second apparatus; and	Rights portion of signed license, received by consumer's computer from "License issuer" (WMRM SDK, Step 9)
(5) hardware or software used for receiving and opening secure containers, said secure containers each including the capacity to contain a governed item, a secure container rule being associated with each of said secure containers;	Windows Media Player and Windows Media Rights Manager
(6) a protected processing environment at least in part protecting information contained in said protected processing environment from tampering by a user of said first apparatus, said protected processing environment including hardware or software used for applying said first secure container rule and a second secure container rule in combination to at least in part govern at least one aspect of access to or use of a governed item contained in a secure container; and	1st and 2nd rules consist of any two valid rules as specified in the Window Media Rights Manager SDK; protected processing environment includes Windows Media Rights Manager and Windows processes for protecting operation of Windows Media Rights Manager. Licenses can be used to convey multiple rules.
(7) hardware or software used for	Any hardware or software employed in

Exhibit B

transmission of secure containers to other apparatuses or for the receipt of secure containers from other apparatuses.	transmitting Windows Media files, including for example consumer's computer's communication port and Windows Media Player (WMRM SDK, Step 3)

Exhibit B

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,185,683

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
2.	Infringing products include Office 2003 and included applications, and Server 2003, including Microsoft hosted RMS Service using Passport
A system including:	
a first apparatus including, user controls, a communications port, a processor, a memory storing:	A device with user controls, a communications port, a processor and memory. For example, the user controls may be a keyboard and mouse, the communications port may be a NIC card with an Ethernet port, the processor may be a CPU, and the memory may be a hard-drive or RAM.
a first secure container containing a governed item, the first secure container governed item being at least in part encrypted; the first secure container having been received from a second apparatus;	An encrypted IRM-governed email received from a remote computer. The encrypted IRM-governed email contains an encrypted IRM-governed email message.
a first secure container rule at least in part governing an aspect of access to or use of said first secure container governed item, the first secure container rule, the first secure container rule having been received from a third apparatus different from said second apparatus; and	The first secure container rule is received from the RMS server in the form of a use license. This use license contains rules generated by the RMS server specifically for the user (or user's group)
hardware or software used for receiving and opening secure containers, said secure containers each including the capacity to contain a governed item, a secure container rule being associated with each of said secure containers;	The RM-enabled device contains hardware or software for receiving and opening secure emails. The secure email has the capacity to contain an IRM-governed email message, with a rule being associated with each email. The rules associated with the secure emails are rules that come as part of the original email as well as rules that come back from the RMS.
a protected processing environment at least in part protecting information contained in said protected processing environment from tampering by a user of said first apparatus, said protected processing environment including hardware or software used for applying said first secure container rule and a second secure container rule in combination to at least in part govern at least one aspect of	Protected information on the RM-enabled device is protected by the use of at least cryptographic techniques. The rule governing the email works together with an additional rule to determine what access to or use (if any) are allowed with respect to the IRM-governed email message. For example, the additional rule may be

access to or use of a governed item contained in a secure container; and	received together with the rule in the use license.
hardware or software used for transmission of secure containers to other apparatuses or for the receipt of secure containers from other apparatuses.	The device includes hardware or software used for transmitting or receiving secure emails. For example, RM-enabled OUTLOOK is designed to transmit and receive encrypted IRM-governed emails to/from other devices.

Exhibit B

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 6,185,683

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
2.	Infringing products include Office 2003 and included applications, and Server 2003, including Microsoft hosted RMS Service using Passport
A system including:	
a first apparatus including, user controls, a communications port, a processor, a memory storing:	A device with user controls, a communications port, a processor, and memory. For example, the user controls may be a keyboard and mouse, the communications port may be a NIC card with an Ethernet port, the processor may be a CPU, and the memory may be a hard-drive or RAM.
a first secure container containing a governed item, the first secure container governed item being at least in part encrypted; the first secure container having been received from a second apparatus;	The first secure container is an encrypted IRM-protected document. This encrypted IRM-governed document is, for example, received from a remote computer, as an attachment to an IRM-governed email or downloaded from a document server or web site.
a first secure container rule at least in part governing an aspect of access to or use of said first secure container governed item, the first secure container rule, the first secure container rule having been received from a third apparatus different from said second apparatus; and	The first secure container rule is received from the RMS server in the form of a use license. This use license contains rules generated by the RMS server specifically for the user (or user's group).
hardware or software used for receiving and opening secure containers, said secure containers each including the capacity to contain a governed item, a secure container rule being associated with each of said secure containers;	The RM-enabled device contains hardware or software for receiving and opening secure documents. The secure documents have the capacity to contain IRM-governed content, with a rule being associated with each secure document. The rules associated with said secure documents are the rules that come as part of the originally received document as well as rules that come back from the RMS server.
a protected processing environment at least in part protecting information contained in said protected processing environment from tampering by a user of said first apparatus,	Protected information on the RM-enabled device is protected by the use of at least cryptographic technique. The rule governing the document works

Exhibit B

said protected processing environment including hardware or software used for applying said first secure container rule and a second secure container rule in combination to at least in part govern at least one aspect of access to or use of a governed item contained in a secure container; and	together with an additional rule to determine what access to or use (if any) are allowed with respect to the IRM-governed document. For example, the additional rule may be associated with an email to which the document was attached, or received together with the rule in the use license.
hardware or software used for transmission of secure containers to other apparatuses or for the receipt of secure containers from other apparatuses.	The device includes hardware or software used for transmitting or receiving secure documents. For example, RM-enabled OUTLOOK is designed to transmit and receive to/from other devices emails with IRM-governed documents attached thereto.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,185,683

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
3.	Infringing products include Office 2003 and included applications, and Server 2003, including Microsoft hosted RMS Service using Passport
A system including:	
a first apparatus including, user controls, a communications port, a processor, a memory storing:	A device with user controls, a communications port; a processor and memory. For example, the user controls may be a keyboard and mouse, the communications port may be a NIC card with an Ethernet port, the processor may be a CPU, and the memory may be a hard-drive or RAM.
a first secure container containing a governed item, the first secure container governed item being at least in part encrypted;	The first secure container containing a governed item is an IRM protected email. Both the email and attachment are IRM protected, each having their own rules, each being encrypted.
a first secure container rule at least in part governing an aspect of access to or use of said first secure container governed item; and	The rule governing the email (a first secure container rule) governs said first secure container governed item.
a second secure container containing a digital certificate;	The second secure container is the IRM protected attachment's derived license request object. The license request object contains the Publishing license and a signed digital certificate.
hardware or software used for receiving and opening secure containers, said secure containers each including the capacity to contain a governed item, a secure container rule being associated with each of said secure containers;	The RM (IRM) enabled computer has software for receiving and opening secure containers. The IRM secure containers have capacity to contain a governed item, with a secure container rule being associated with each of said secure containers.
a protected processing environment at least in part protecting information contained in said protected processing environment from tampering by a user of said first apparatus, said protected processing environment including hardware or software used for	Protected information on the RM-enabled computer is protected by the use of at least cryptographic techniques. The rules governing the email itself (first

applying said first secure container rule and a second secure container rule in combination to at least in part govern at least one aspect of access to or use of a governed item contained in a secure container; and	secure container rule) and the rules governing the attachment work together to determine what access to or use (if any) will be allowed with respect to the governed item.
hardware or software used for transmission of secure containers to other apparatuses or for the receipt of secure containers from other apparatuses.	IRM-enabled applications, e.g., OUTLOOK, are designed to transmit and receive RM secured containers to/from other computers.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,185,683

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
3.	Infringing products include Office 2003 and included applications, and Server 2003, including Microsoft hosted RMS Service using Passport
A system including:	
a first apparatus including, user controls, a communications port, a processor, a memory storing:	A device with user controls, a communications port, a processor and memory. For example, the user controls may be a keyboard and mouse, the communications port may be a NIC card with an Ethernet port, the processor may be a CPU, and the memory may be a hard-drive or RAM.
a first secure container containing a governed item, the first secure container governed item being at least in part encrypted;	The first secure container containing a governed item is an IRM protected document, which is an attachment within an IRM protected email message. The governed item is the document's content. Both the email message and attachment are encrypted and have associated usage rules due to IRM protection.
a first secure container rule at least in part governing an aspect of access to or use of said first secure container governed item; and	A use license for the IRM protected document specifies rules governing access to or use of said first secure container governed item.
a second secure container containing a digital certificate;	The second secure container is the IRM protected email message. The IRM protected attachment includes a publishing license and an owner certificate, both of which are signed XrML digital certificates. The attachment (including embedded certificates) is contained within the IRM protected email message (said second secure container).
hardware or software used for receiving and opening secure containers, said secure containers each including the capacity to contain a governed item, a secure container rule being associated with each of said secure containers;	The RM (IRM) enabled computer has software for receiving and opening secure containers. The IRM secure containers have capacity to contain a governed item, with a secure container rule being associated with each of said secure containers.
a protected processing environment at least in part protecting information contained in said protected processing environment from	Protected information on the RM-enabled computer is protected by the use of at least cryptographic techniques.

293482.02

PAGE 84/92 * RCVD AT 8/4/2004 8:16:32 PM [Eastern Daylight Time] * SVR:USPTO-EFXRF-1/1 * DNIS:8729306 * CSID:6508496775 * DURATION (mm-ss):28-08

tampering by a user of said first apparatus, said protected processing environment including hardware or software used for applying said first secure container rule and a second secure container rule in combination to at least in part govern at least one aspect of access to or use of a governed item contained in a secure container; and	The rules governing the attachment (first secure container rule) and the rules governing the email message (second secure container rule) work together to determine what access to or use (if any) will be allowed with respect to the governed item.
hardware or software used for transmission of secure containers to other apparatuses or for the receipt of secure containers from other apparatuses.	RM-enabled applications, e.g., OUTLOOK, are designed to transmit and receive RM secured containers to/from other computers.
4. A system as in claim 3,	
said memory storing a rule associated with said second secure container, said rule associated with said second secure container at least in part governing at least one aspect of access to or use of said digital certificate.	All parts of the attachment (including embedded signed XrML licenses/certificates) are protected by the enclosing email message and governed by the associated email rules (second secure container rule).

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,185,683

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
5.	Infringing products include Office 2003 and included applications, and Server 2003, including Microsoft hosted RMS Service using Passport
A system including:	
a first apparatus including, user controls, a communications port, a processor, a memory storing:	A device with user controls, a communications port, a processor and memory. For example, the user controls may be a keyboard and mouse, the communications port may be a NIC card with an Ethernet port, the processor may be a CPU, and the memory may be a hard-drive or RAM.
a first secure container containing a governed item, the first secure container governed item being at least in part encrypted;	first secure container containing a governed item is an IRM protected email. Both the email and attachment are IRM protected, each having their own rules, each being encrypted.
a first secure container rule at least in part governing an aspect of access to or use of said first secure container governed item; and	The rule governing the email (a first secure container rule) governs said first secure container governed item.
a second secure container containing a digital signature, the second secure container being different from said first secure container;	The second secure container is the IRM protected attachment's derived license request object. The license request object contains the Publishing license and a signed digital certificate.
hardware or software used for receiving and opening secure containers, said secure containers each including the capacity to contain a governed item, a secure container rule being associated with each of said secure containers;	The RM (IRM) enabled computer has software for receiving and opening secure containers. The IRM secure containers have capacity to contain a governed item, with a secure container rule being associated with each of said secure containers.
a protected processing environment at least in part protecting information contained in said protected processing environment from tampering by a user of said first apparatus, said protected processing environment including hardware or software used for applying said first secure container rule and a	Protected information on the RM-enabled computer is protected by the use of at least cryptographic techniques. The rules governing the email itself (first secure container rule) and the rules governing

second secure container rule in combination to at least in part govern at least one aspect of access to or use of a governed item contained in a secure container; and	the attachment will work together to determine what access to or use (if any) will be allowed with respect to the governed item.
hardware or software used for transmission of secure containers to other apparatuses or for the receipt of secure containers from other apparatuses.	RM-enabled applications, e.g., OUTLOOK, are designed to transmit and receive RM secured containers to/from other computers.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,185,683

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
5.	Infringing products include Office 2003 and included applications, and Server 2003, including Microsoft hosted RMS Service using Passport
A system including:	
a first apparatus including, user controls, a communications port, a processor, a memory storing:	A device with user controls, a communications port, a processor and memory. For example, the user controls may be a keyboard and mouse; the communications port may be a NIC card with an Ethernet port, the processor may be a CPU, and the memory may be a hard-drive or RAM.
a first secure container containing a governed item, the first secure container governed item being at least in part encrypted;	first secure container containing a governed item is an IRM protected email. Both the email and attachment are IRM protected, each having their own rules, each being encrypted.
a first secure container rule at least in part governing an aspect of access to or use of said first secure container governed item; and	The rule governing the email (a first secure container rule) governs said first secure container governed item.
a second secure container containing a digital signature, the second secure container being different from said first secure container;	The second secure container is the IRM email attachment. This attachment and its publishing license are signed.
hardware or software used for receiving and opening secure containers, said secure containers each including the capacity to contain a governed item, a secure container rule being associated with each of said secure containers;	The RM (IRM) enabled computer has software for receiving and opening secure containers. The IRM secure containers have capacity to contain a governed item, with a secure container rule being associated with each of said secure containers.
a protected processing environment at least in part protecting information contained in said protected processing environment from tampering by a user of said first apparatus, said protected processing environment including hardware or software used for applying said first secure container rule and a	Protected information on the RM-enabled computer is protected by the use of at least cryptographic techniques. The rules governing the email itself (first secure container rule) and the rules governing

second secure container rule in combination to at least in part govern at least one aspect of access to or use of a governed item contained in a secure container; and	the attachment work together to determine what access to or use (if any) will be allowed with respect to the governed item.
hardware or software used for transmission of secure containers to other apparatuses or for the receipt of secure containers from other apparatuses.	RM-enabled applications, e.g., OUTLOOK,are designed to transmit and receive RM secured containers to/from other computers.

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP. INTERTRUST INFRINGEMENT CHART FOR U.S. PATENT NO. 6,185,683

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
5.	Infringing products include Office 2003 and included applications, and Server 2003, including Microsoft hosted RMS Service using Passport
A system including:	
a first apparatus including, user controls, a communications port, a processor, a memory storing:	A device with user controls, a communications port, a processor and memory. For example, the user controls may be a keyboard and mouse, the communications port may be a NIC card with an Ethernet port, the processor may be a CPU, and the memory may be a hard-drive or RAM.
a first secure container containing a governed item, the first secure container governed item being at least in part encrypted;	The first secure container containing a governed item is an IRM protected document, which is an attachment within an IRM protected email message. The governed item is the document's content. Both the email message and attachment are encrypted and have associated usage rules due to IRM protection.
a first secure container rule at least in part governing an aspect of access to or use of said first secure container governed item; and	A use license for the IRM protected document specifies rules governing access to or use of said first secure container governed item.
a second secure container containing a digital signature, the second secure container being different from said first secure container;	The second secure container is the IRM protected email message. The IRM protected attachment includes a publishing license and an owner certificate, both of which are signed XrML digital certificates. The attachment (including embedded certificates) is contained within the IRM protected email message (said second secure container).
hardware or software used for receiving and opening secure containers, said secure containers each including the capacity to contain a governed item, a secure container rule being associated with each of said secure containers;	The RM (IRM) enabled computer has software for receiving and opening secure containers. The IRM secure containers have capacity to contain a governed item, with a secure container rule being associated with each of said secure containers.
a protected processing environment at least in part protecting information contained in said	Protected information on the RM-enabled computer is protected by the use of at least

protected processing environment from tampering by a user of said first apparatus, said protected processing environment including hardware or software used for applying said first secure container rule and a second secure container rule in combination to at least in part govern at least one aspect of access to or use of a governed item contained in a secure container; and	cryptographic techniques. The rules governing the attachment (first secure container rule) and the rules governing the email message (second secure container rule) work together to determine what access to or use (if any) will be allowed with respect to the governed item.
hardware or software used for transmission of secure containers to other apparatuses or for the receipt of secure containers from other apparatuses.	RM-enabled applications, e.g., OUTLOOK, are designed to transmit and receive RM secured containers to/from other computers.
6. A system as in claim 5,	
said memory storing a rule at least in part governing an aspect of access to or use of said digital signature.	All parts of the attachment (including embedded signed XrML licenses/certificates) are protected by the enclosing email message and governed by the associated email rules (second secure container rule).

Exhibit B

INTERTRUST TECHNOLOGIES CORP. v. MICROSOFT CORP.
INTERTRUST INFRINGEMENT CHART
FOR U.S. PATENT NO. 6,185,683

CLAIM LANGUAGE	CLAIM OF INFRINGEMENT
28.	Infringing products include Office 2003 and included applications, and Server 2003, including Microsoft hosted RMS Service using Passport
A system including:	
a first apparatus including; user controls, a communications port, a processor, a memory containing a first rule,	A device with user controls, a communications port, a processor and memory. For example, the user controls may be a keyboard and mouse, the communications port may be a NIC card with an Ethernet port, the processor may be a CPU, and the memory may be a hard-drive or RAM. The first rule governs use of an IRM protected document (e.g., an IRM rule permitting a document to be read by specified users or barring access to IRM-governed information from specified users, applications, or other principals).
hardware or software used for receiving and opening secure containers, said secure containers each including the capacity to contain a governed item, a secure container rule being associated with each of said secure containers;	The RM-enabled device contains hardware or software for receiving and opening secure containers. The secure email has the capacity to contain an IRM-governed email message, with a rule being associated with each email.
a protected processing environment at least in part protecting information contained in said protected processing environment from tampering by a user of said first apparatus, said protected processing environment including hardware or software used for applying said first rule and a secure container rule in combination to at least in part govern at least one aspect of access to or use of a governed item; and	Protected information on the RM-enabled device is protected by the use of at least cryptographic techniques. The secure container rule is an IRM rule governing access to the IRM protected document (e.g., a rule permitting editing by specified users). The rule governing the email works together with an additional rule to determine what access to or use (if any) are allowed with respect to the IRM-governed email message (the document's content). For example, the additional rule may be received together with the rule in the use license, may be associated with a publishing license, may be associated with user certification, revocation lists, or exclusion policies, or may be received from any other source.
hardware or software used for transmission of	The device includes hardware or software used